Amendments To The Specification:

Please amend the first full paragraph on page 3 as follows:

In the software copy protection method described in ~~[4]~~ **DE 3914233**, a crypto function is calculated in a connector assembly connected to the PC and parallel to this in the protected application. Partial functions of this crypto function can be inserted into the application on different positions, so that extraction is not possible without a semantic analysis of the program code. With the aid of the output values of the crypto function, calculations of the software are made erroneous and corrected with the aid of the output values of the connector assembly shortly before they can have harmful effects on the course of the application. Without the connector assembly, which is not reproducible for the aggressor, the application can not be used. The described method has the disadvantage that the integration of the partial functions into the software which is to be protected is very laborious.

Please amend the second full paragraph on page 3 as follows:

In a further copy protection method, parts of the program which is to be protected are kept in a not readable memory of a smartcard and are executed by the smartcard controller. The transmission of these parts takes place. only in an encoded manner. Examples of such processors are devices in the form of USB-apparatuses of the companies Syncrosoft ~~[12]~~ and Sospita ~~[11]~~. The encryption of the software in this method prevents reverse engineering, too.

Please amend the first full paragraph on page 4 as follows:

In ~~[10]~~ **US 6192475**, a method against reverse engineering of software is described, which cloaks logical connections between elementary operations of the processor and data streams by the introduction of complex addressing mechanisms. One disadvantage of the invention is shown in the attempt to protect object-oriented software. In general, software that had been developed in an object-oriented manner contains very short methods, which are constituted by a small number of program instructions and realise in most cases very simple data streams with a small number of variables. At least in this case, the described method is not effective. Further, no possibility is known for this method which produces a hardly detachable linkage to a hardware and thus prevents copying of the transformed software.

Please amend the first full paragraph on page 5 as follows:

According to claim 1, a Petri net is encoded, the transitions of which exchange symbols or symbol strings with the aid of at least one or plural heads with at least one tape. The encoding of the Petri net is written into a memory and read and executed by at least one instance. Petri nets and the terms "position", "transition" and "mark" are described in **E. Jessen and R. Valk, *Rechensysteme, Grundlagen der Modellbidung* (1987); and W. Reisig, *Petri-Netze, Eine Einführung* (1982)** **~~[6] and [8]~~.** The terms "head" and "tape" are used in conformity with the terms describing a Turing machine, the tape being a finite one for technical reasons, in difference to the model of the Turing machine. Turing machines are described in **J. E. Hopcroft and J.D. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie,* (4th ed. 2000)** ~~[5]~~, for instance. Preferably, the head is moved on the tape at each reading and writing operation. However, the movement of the head may also be controllable. Further, the existence of at least two heads is advantageous for the operation speed, because most operations work with at least two operands. A tape may be a register of a processor or a memory cell of a RAM. A head may be a register with a mask for the masking of values of the tape. With the execution of a Petri net, the switching of transitions of the Petri net is to be understood here. By the execution of the Petri net, which works on tapes, data are processed. The memory and the executing instance or the executing instances, respectively, can be realised in many ways. For the concept of the invention, it is important that the semantics which is behind the Petri net is difficult to analyse even when the Petri net is known. Preferably, the generation and the encoding of the Petri net take place in a memory different from that one for the execution. The encoding of the Petri net, the heads, tapes, fields and symbols is possible in many variants. An aggressor who wants to gather information about the semantics of the Petri net has only the possibility to compare the Petri net with those Petri nets he already knows, or to guess the semantics with the aid of input and output examples. According to claim 20, the Petri net can receive and process symbols or symbol strings from a cryptological function. The cryptological function can be fixedly attached to the device which executes the Petri net, so that a linkage of the processing method with a hardware is created which is difficult to detach for an aggressor.

Please amend the third full paragraph on page 5 as follows:

According to a further embodiment of the invention, the switching of transitions can be rapidly performed with the aid of tables. In analogy to sequential machines, as described in ~~[2]~~ **T. L. Booth, *Sequential Machines and Automata Theory* (1967),** for instance, a derived mark or a derived state, respectively, and an output may be rapidly determined from a table on the basis of a mark or a state, respectively, and an input. The inputs or outputs, respectively, may also take place optionally.

Please amend the last paragraph on page 7 as follows:

An alternative solution of the objective on which the invention is based provides according to claim 9 that data-processing, co-operating nets are composed, the composition result is encoded, written into a memory and read and executed from the memory by at least one instance, wherein the composition result is a net which is equivalent to its components with respect to input/output behaviour, except output delays. Exempted from this is a public key encryption method of ~~[1]~~ **Feng Bao and Yoshihide Igarashi, *Break Finite Automata Public Key Cryptosystem*, ICALP 1995: 147-158 (1995)** and ~~[3]~~ **Zongduo Dai, Dingfeng Ye, Kwok-Yan Lam, *Weak Invertibility of Finite Automata and Cryptoanalysis on FAPKC*, Advances in Cryptology – ASIACRYPT'98: 227-241 (1998)**, in which the composition result of a composition of finite automates form a public key. In the present invention, it is dealt with the general processing of data, taking into account the objective on which the present invention is based. The objective is resolved because a semantic analysis of a composition result is difficult without knowing the components. In many cases, a decomposition is a hard problem or np-hard problem, respectively.

Please amend the first full paragraph on page 8 as follows:

The feature of claim 9 does not delimit which kinds of data-processing, co-operating nets are composed. It is known that many nets of the one kind can be simulated by nets of another kind or are equivalent to each other, respectively. For instance, in ~~[7]~~ **M. Minsky, *Computation: Finite and Infinite Machines* (1967)** and ~~[9]~~ **R. Rojas, *Theorie der neuronalen Netze* (1993),** it has been shown that recursive McCulloch-Pitts nets, a special form of artificial neuronal nets, are equivalent to finite automates. Finite automates can be described by B/E nets again. B/E nets are special Petri nets. As is naturally, any description of the composition depends on the formal definition of the nets, and many variants of the composition that differ in content can be defined, irrespective of this definition. Claim 9 also includes variants of compositions which are based on the same concept of the invention.

Please amend the paragraph on page 9 as follows:

A set of synchronization channels is a parameter of the composition function. The transitions of the machine which is to be composed switch depending from an imaginary global clock and there is no by-passing. A "rendez-vous" between sender and receiver of symbols should be possible, which requires that the components can wait for each other. This is realised by switching an "empty transition" of the waiting machine. The empty transition does not read anything nor does it write anything. Such transitions exist in non-deterministic automates with λ-movements ~~[5]~~. The λ -movements are called ε-movements here. In the non-deterministic sequential machines to be composed as B/E nets, there are plural possible switching sequences or serial processes ~~[6]~~. Each possible switching sequence corresponds to one composed sequential machine. The composition unction is a mapping in a power set of sequential machines. Let be $\Omega = (C, \Delta., \gamma)$ a communication rule and B with $B \subseteq C$ a set of internal synchronization channels. The composition $comp_B : M^n_\Omega \rightarrow 2^{M\Omega}$ is defined as:

Please amend the third full paragraph on page 12 as follows:

After a composition, undesired compositions with empty sets of events are often generated. These transitions can be replaced by a position-edged roughening (explication of this term in ~~[6]~~ **E. Jessen and R. Valk, *Rechensysteme, Grundlagen der Modellbidung*, (1987)**), wherein the edges are the entrance- and exit positions of the transition. This is repeated so often until there are no more empty transitions in the machine. This mapping is designated by a function *min*: $M_\Omega \rightarrow M_\Omega$ in the following.

Please amend the first full paragraph on page 16 as follows:

With the aid of an alternative method, it is achieved that the execution of a data-processing net or program, respectively, is coupled to the executing device. A cryptological function, which is executed in a protected way, e.g. a function of the TPM-chip of the Trusted Computing Platform Alliance (TCPA) ~~[13]~~, which is fixedly attached to the device, a PC or a PDA for instance, exchanges data with the net or the program, respectively. The data-processing net or the program, respectively, do not work or work erroneously when no or erroneous data are received from the cryptological function. In one form of realisation of the method, a value exceeding a calculation of the function value of the cryptological function is stored in a fashion which makes it not readable or changeable for an aggressor, and in a following calculation of a further function value, this value influences the result of the following calculation, wherein this value changes according to a predetermined rule. Through this, it is prevented that plural net instances or program instances, respectively, can use uncontrolled function values of the cryptological function.

Please amend the paragraph beginning on page 22 as follows:

The components form a closed circuit of symbol producers and symbol consumers. After the composition, the head is positioned on field F_3, which memorises a one. A finite control P_1 for recognising the language $L = \{0^n 1^n \mid n \geq 1\}$ is represented in FIG. 23. It is presupposed that the word which is to be recognised stands flush left on the tape with a fore-going and a following τ, and that the head is positioned on a field on the left of the τ at the right side. Supposed the word belongs to language L, there is the following procedure (similar to that in ~~[5]~~ **J. E. Hopcroft and J.D. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, (4th ed. 2000)**): The one standing farthest at right is replaced by τ. Thereafter, the head moves towards the left side up to the next τ, and then one field to the right side. The zero standing farthest at left is memorised here, and is replaced by τ. Thereafter, the one standing farthest at the right and then the zero standing farthest on the left is replaced by τ, and so forth. In the case that a zero was replaced by τ and a τ was found next to it on the right side, the machine accepts the word. The accepting is communicated to the outer world by writing a one on the channel Λ. If the machine, when seeking for a zero or a one, respectively, finds a one or a zero, respectively, or a τ ,the word is not accepted and a zero is outputted on channel Λ. The user (the finite control of the tape in this case) does not need to have any knowledge about the structure of the tape. The band and the fields might also be composed in one machine, for instance. The user must only have knowledge about the meaning of the input and output channels, via the interface of the tape. A user of the finite control for recognising the language has to know how the word that is to be examined must be written on the tape. To this belongs the knowledge about the meaning of channel I and the convention to write a foregoing τ on the first field of the tape. This knowledge may be encapsulated by the machine in FIG. 24. When this encapsulation is composed with the tape, symbol strings in the form as shown in FIG. 25 can be inputted. Let be $c \in L$. When E is concatenated with P_1 (ending state of E is the starting state of P_1) and this is composed with T, all channels used by at least two machines being synchronisation channels, one gets a machine after applying *red* which is represented in FIG. 26 a). When $c \notin L$, the result is the machine represented in FIG. 26 b).

Please amend the paragraph beginning at page 25 as follows:
Sequential reversible machines can be used for encoding and decoding. One example for a sequential, reversible machine is represented in FIG. 32. In every state, at given output the input belonging to it can be unambiguously determined. Such machines with a significantly higher number of states than exemplified in FIG. 32 are suited for the composition with other nets, as is shown in FIGS. 27 a) and b), for instance. Machines with delayed outputs, as described in **Zongduo Dai, Dingfeng Ye, Kwok-Yan Lam, *Weak Invertibility of Finite Automata and***

***Cryptoanalysis on FAPKC*, Advances in Cryptology – ASIACRYPT'98: 227-241 (1998)**, are also conceivable. All the machines can be generated in a non-deterministic manner, for instance with the aid of random number generators.